

## **Oracle ADF interview Question Part – 1**

**Read more:** <http://www.techartifact.com/blogs/2011/04/oracle-adf-interview-question-part-1.html#ixzz2Buy7WMHl>

**What is Oracle Adf?**

**Ans :**

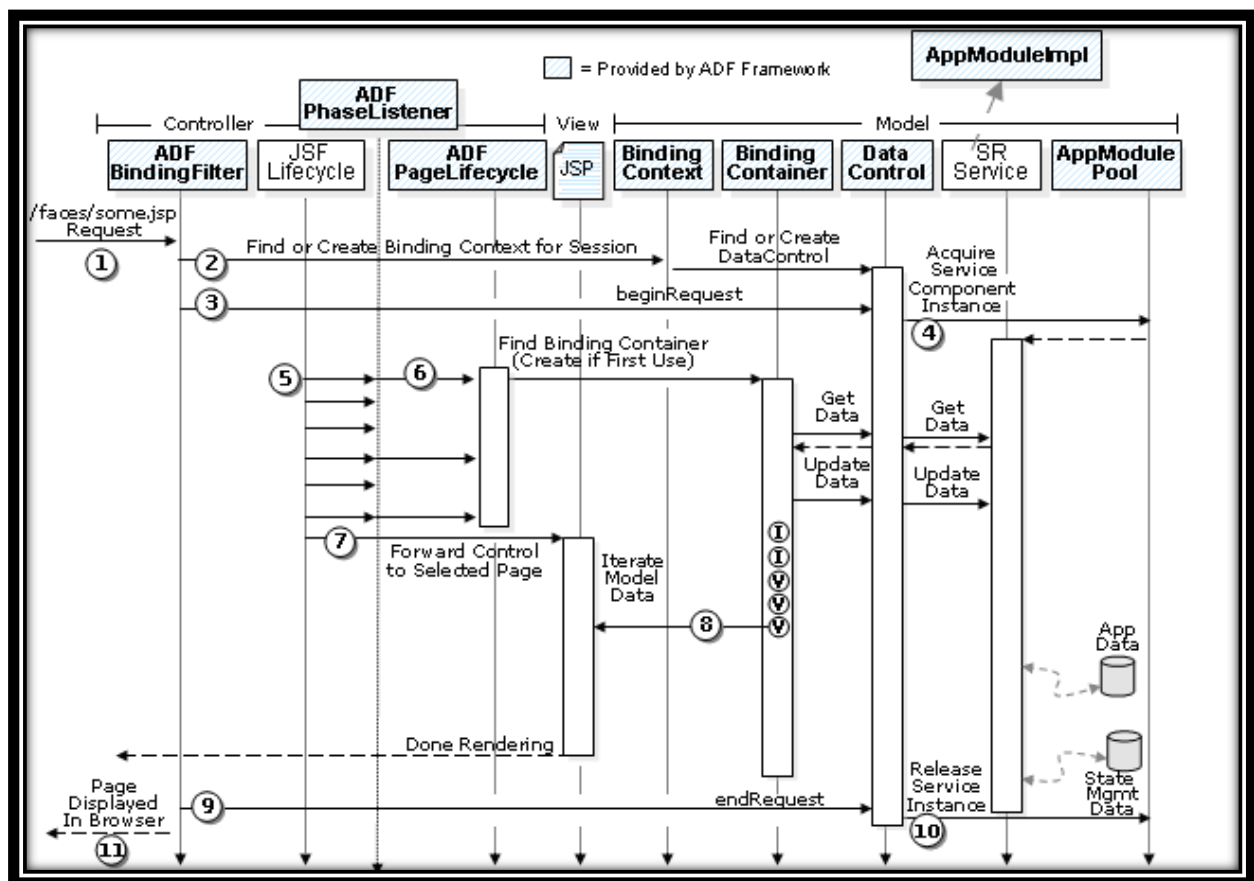
The Oracle Application Development Framework (Oracle ADF) is an end-to-end application framework that builds on J2EE standards and open-source technologies to simplify and accelerate implementing service-oriented applications. If you develop enterprise solutions that search, display, create, modify, and validate data using web, wireless, desktop, or web services interfaces, Oracle ADF can simplify your job. Used

in tandem, Oracle JDeveloper 10g and Oracle ADF give you an environment that covers the full development lifecycle from design to deployment, with drag-and-drop data binding, visual UI design, and team development features built-in.

**Q: Lifecycle of a Web Page Request Using Oracle ADF and JSF**

**Ans :** Below figure shows a sequence diagram of the lifecycle of a web page request using JSF and Oracle ADF in tandem.

**Lifecycle of a Web Page Request Using JSF and Oracle ADF**



As shown in the figure, the basic flow of processing happens as follows:

A web request for `http://yourserver/yourapp/faces/some.jsp` arrives from the client to the application server

The ADFBindingFilter finds the ADF binding context in the HTTP session, and if not yet present, initializes it for the first time.

During binding context initialization, the ADFBindingFilter:

Consults the servlet context initialization parameter named `CpxFileName` and appends the `*.cpx` file extension to its value to determine the name of the binding context metadata file. By default the parameter value will be `"DataBindings"`, so it will look for a file named `DataBindings.cpx`.

Reads the binding context metadata file to discover the data control definitions, the page definition file names used to instantiate binding containers at runtime, and the page map that relates a JSP page to its page definition file.

Constructs an instance of each Data Control, and a reference to each BindingContainer. The contents of each binding container are loaded lazily the first time they are used by a page.

The ADFBindingFilter invokes the beginRequest() method on each data control participating in the request. This gives every data control a notification at the start of every request where they can perform any necessary setup.

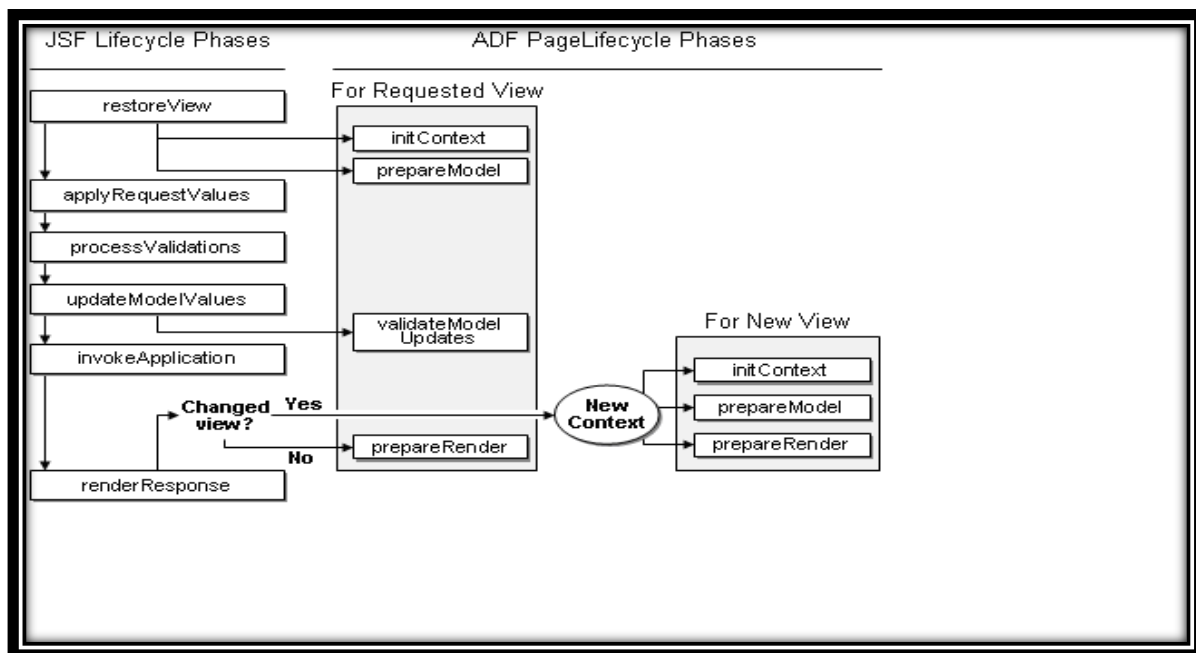
An application module data control uses the beginRequest notification to acquire an instance of the application module from the application module pool.

The JSF Lifecycle class, which is responsible for orchestrating the standard processing phases of each request, notifies the ADFPhaseListener class during each phase of the lifecycle so that it can perform custom processing to coordinate the JSF lifecycle with the Oracle ADF Model data binding layer.

Note: The FacesServlet (in javax.faces.webapp) is configured in the web.xml file of a JSF application and is responsible for initially creating the JSF Lifecycle class (in javax.faces.lifecycle) to handle each request. However, since it is the Lifecycle class that does all the interesting work, the FacesServlet is not shown in the diagram.

The ADFPhaseListener creates an ADF PageLifecycle object to handle each request and delegates appropriate before/after phase methods to corresponding methods in the ADF PageLifecycle class as shown in . If the appropriate binding container for the page has never been used before during the user's session, it is created.

### How JSF Page Lifecycle and ADF Page Lifecycle Phases Relate



The JSF Lifecycle forwards control to the page to be rendered.

The UI components on the page access value bindings and iterator bindings in the page's binding container and render the formatted output to appear in the browser.

The ADFBindingFilter invokes the end Request() method on each data control participating in the request. This gives every data control a notification at the end of every request where they can perform any necessary resource cleanup.

An application module data control uses the endRequest notification to release the instance of the application module back to the application module pool.

The user sees the resulting page in the browser.

### **Q : What is Action Listener ?**

**Ans :** An action listener is a class that wants to be notified when a command component

fires an action event. An action listener contains an action listener method that processes the action event object passed to it by the command component

### **Q:What are business Component In ADF.Describe them?**

Ans: All of these features can be summarized by saying that using ADF Business Components for your J2EE business service layer makes your life a lot easier. The key

ADF Business Components components that cooperate to provide the business service

implementation are:

#### **■ *Entity Object***

An entity object represents a row in a database table and simplifies modifying its data by handling all DML operations for you. It can encapsulate business logic for the row to ensure your business rules are consistently enforced. You associate an entity object with others to reflect relationships in the underlying database schema to create a layer of business domain objects to reuse in multiple applications.

### ■ *Application Module*

An application module is the transactional component that UI clients use to work with application data. It defines an updatable data model and top-level procedures and functions (called service methods) related to a logical unit of work related to an end-user task.

### ■ *View Object*

A view object represents a SQL query and simplifies working with its results. You use the full power of the familiar SQL language to join, project, filter, sort, and aggregate data into exactly the “shape” required by the end-user task at hand. This includes the ability to link a view object with others to create master/detail hierarchies of any complexity. When end users modify data in the user interface, your view objects collaborate with entity objects to consistently validate and save the changes.

### **Q: What is Top Link?**

**Ans:**

Top Link is an Object-Relational Mapping layer that provides a map between the Java objects that

the model uses and the database that is the source of their data.

By default, a session is created named **default**. In the following steps, you create a new session

### **Q:What is Managed Bean?**

**Ans:**JavaBean objects managed by a JSF implementation are called managed beans. A managed bean describes how a bean is created and managed. It has nothing to do with the bean’s functionality.

**Managed** bean is about how the bean is created and initialized. As you know, jsf uses the lazy initialization model. It means that the bean in the particular scope is created and initialized not at the moment when the scope is started, but on-demand, i.e. when the bean is first time required.

### **Q :What is Backing Bean?**

**Ans:**Backing beans are JavaBeans components associated with UI components used in a page. Backing-bean management separates the definition of UI component objects from objects that perform application-specific processing and hold data.

**Backing** bean is about the role a particular managed bean plays. This is a role to be a server-side representation of the components located on the page. Usually, the backing beans have a request scope, but it is not a restriction.

The backing bean defines properties and handling-logics associated with the UI components used on the page. Each backing-bean property is bound to either a component instance or its value. A backing bean also defines a set of methods that perform functions for the component, such as validating the component's data, handling events that the component fires and performing processing associated with navigation when the component activates.

### **What is view object?**

**Ans :**A view object is a model object used specifically in the presentation tier. It contains the data that must display in the view layer and the logic to validate user input, handle events, and interact with the business-logic tier. The backing bean is the view object in a JSF-based application. Backing bean and view object are interchangeable terms

## Q: Difference between Backing Bean and Managed Bean?

Backing Beans	Managed Beans
A backing bean is any bean that is referenced by a form.	A managed bean is a backing bean that has been registered with JSF (in faces-config.xml) and it automatically created (and optionally initialized) by JSF when it is needed.
	The advantage of managed beans is that the JSF framework will automatically create these beans, optionally initialize them with parameters you specify in faces-config.xml,
Backing Beans should be defined only in the request scope	The managed beans that are created by JSF can be stored within the request, session, or application scopes

Q? What do you mean by Bean Scope?

Bean Scope typically holds beans and other objects that need to be available in the different components of a web application.

## Q. What are the different kinds of Bean Scopes in JSF?

JSF supports three Bean Scopes. *viz.*,

**Request Scope:** The request scope is short-lived. It starts when an HTTP request is submitted and ends when the response is sent back to the client.

**Session Scope:** The session scope persists from the time that a session is established until session termination.

**Application Scope:** The application scope persists for the entire duration of the web application. This scope is shared among all the requests and sessions.

## What is the difference between JSP-EL and JSF-EL?

JSP-EL	JSF-EL
In JSP-EL the value expressions are delimited by \${...}.	In JSf-EL the value expressions are delimited by #{...}.
The \${...} delimiter denotes the	The #{...} delimiter denotes deferred

immediate evaluation of the expressions, at the time that the application server processes the page.

evaluation. With deferred evaluation, the application server retains the expression and evaluates it whenever a value is needed.

**Q:How to declare the page navigation (navigation rules) in faces-config.xml file in ADF 10g?**

**Ans:** Navigation rules tells JSF implementation which page to send back to the browser after a form has been submitted. We can declare the page navigation as follows:

```
<naviagation-rule>
<from-view-id>/index.jsp</from-view-id>
<navigation-case>
<from-outcome>login</from-outcome>
<to-view-id>/welcome.jsp</to-view-id>
</navigation-case>
</naviagation-rule>
```

This declaration states that the *login* action navigates to */welcome.jsp*, if it occurred inside */index.jsp*.

**Q: Setting the range of table**

Ans : <af:table rows="#{bindings.LoggedInUserServiceRequests.rangeSize}".../>

**Q : Which component in ADF BC manages transaction ?**

A : Application Module, manages transaction.

**Q : Can an entity object be based on two Database Objects(tables/views) or two Webservices ?**

A : No entity objects will always have one to one relationship with a database object or web service.



**Q : Where is that we write business rules/validations in ADF and why?**

A : We should ideally be writing validations at Entity Object level, because they provide highest degree of reuse.

**Q:What are the JSF life-cycle phases?**

Ans:The six phases of the JSF application lifecycle are as follows (note the event processing at each phase):

1. Restore view
2. Apply request values; process events
3. Process validations; process events
4. Update model values; process events
5. Invoke application; process events
6. Render response

**Q. Explain briefly the life-cycle phases of JSF?**

1. **Restore View** : A request comes through the FacesServlet controller. The controller examines the request and extracts the view ID, which is determined by the name of the JSP page.
2. **Apply request values**: The purpose of the apply request values phase is for each component to retrieve its current state. The components must first be retrieved or created from the FacesContext object, followed by their values.
3. **Process validations**: In this phase, each component will have its values validated against the application's validation rules.
4. **Update model values**: In this phase JSF updates the actual values of the server-side model ,by updating the properties of your backing beans.
5. **Invoke application**: In this phase the JSF controller invokes the application to handle Form submissions.
6. **Render response**: In this phase JSF displays the view with all of its components in their current state.

Read more here

<http://www.techartifact.com/blogs/2012/08/q-explain-briefly-the-life-cycle-phases-of-jsf.html>

### Q: What is setActionListener?

**Ans: SetActionListener** – The setActionListener tag is a declarative way to allow an action source ( , , etc.) to set a value before navigation. It is perhaps most useful in conjunction with the “processScope” EL scope provided by ADF Faces, as it makes it possible to pass details from one page to another without writing any Java code. This tag can be used both with ADF Faces commands and JSF standard tags.

Example of this can be as follows. Suppose we have a table “employee”. We want to fetch the salary of an employee of some particular row and want to send this salary in Next page in process scope or request scope etc. So using this we can do this.

It has two attributes :

From – the source of the value; can be an EL expression or a constant value

To – the target for the value; must be an EL expression

```
<af:setActionListener  
1  from="#{row.salary}"  
  
2  to="#{processScope.salary1}"/>
```

This setActionListener will pick value of salary of that row and store this value into salary1 variable. So anyone can use this salary

As processScope.salary1 . It is very simple to use. And very useful.

Read more: <http://www.techartifact.com/blogs/2011/04/oracle-adf-interview-question-part-1.html#ixzz2BuxZe3av>